UNITED STATES PATENT APPLICATION

for

# A QUEUE PARTITIONING MECHANISM

Inventors:

Sarath Kotamreddy
Tuong Trieu

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(303) 740-1980

File No.: 042390.P19124

"Express Mail" mailing label number EV306655177US
Date of Deposit March 8, 2004
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, P.O. Box 1450, Alexandria, VA 22313-1450

Leah Schwenke
(Typed or printed name of person mailing paper or fee)

Leah Schwenke
(Signature of person mailing paper or fee)

# A QUEUE PARTITIONING MECHANISM

## FIELD OF THE INVENTION

**[0001]** The present invention relates to computer systems; more particularly, the present invention relates to interfacing computer system chipsets.

## BACKGROUND

**[0002]** Integrated Graphics chipsets typically include a graphics accelerator and a memory controller. The graphics accelerator includes a 2D/3D instruction processing unit to control the 2D and 3D graphics engines. These graphics engines interact with a main memory device through the memory controller. The instructions to the memory are carried out through certain command requests, which are processed through a queuing mechanism. The queuing mechanism is used to store some of the information from the graphics engines prior to the information being presented to the memory.

**[0003]** As the size of integrated circuit dies have increased it has become necessary to split up a die into different partitions in order to fulfill the constraints of various back-end tools. Back-end is the process where the die logic is synthesized (e.g., using Synopsys), and goes through layout for auto place and route (APR), after which parasitic extraction and delay calculation are implemented to determine inter-connect delays and the delays through various gates. This extracted information is then used to determine the performance validation (PV) timings using Prime Time. This process is further complicated by the fact that the operating frequencies are also increasing.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention. The drawings, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0005] **Figure 1** illustrates one embodiment of a computer system;

[0006] **Figure 2** illustrates one embodiment of a queue partitioning mechanism; and

[0007] **Figure 3** illustrates another embodiment of a queue partitioning mechanism.

## DETAILED DESCRIPTION

**[0008]** A queue partitioning mechanism is described. Reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

**[0009]** In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention

**[0010]** **Figure 1** is a block diagram of one embodiment of a computer system 100. Computer system 100 includes a central processing unit (CPU) 102 coupled to bus 105. In one embodiment, CPU 102 is a processor in the Pentium® family of processors including the Pentium® II processor family, Pentium® III processors, and Pentium® IV processors available from Intel Corporation of Santa Clara, California. Alternatively, other CPUs may be used.

**[0011]** A chipset 107 is also coupled to bus 105. Chipset 107 includes a memory control hub (MCH) 110. In one embodiment, MCH 110 is coupled to an input/output control hub (ICH) 140 via a hub interface. ICH 140 provides an interface to input/output (I/O) devices within computer system 100. For instance, ICH 140 may be coupled to a Peripheral Component Interconnect bus adhering to a Specification Revision 2.1 bus developed by the PCI Special Interest Group of Portland, Oregon.

**[0012]** In one embodiment, MCH 110 includes a memory controller 112 that is coupled to a main system memory 115. Main system memory 115 stores data and sequences of instructions and code represented by data signals that may be executed by CPU 102 or any other device included in system 100. In one embodiment, main system memory 115 includes dynamic random access memory (DRAM); however, main system memory 115 may be implemented using other memory types.

**[0013]** MCH 110 also includes a graphics accelerator 113 to compute graphical transformations. In one embodiment, graphics accelerator 113 includes a 2D/3D instruction processing unit to control 2D and 3D graphics engines. The 2D and 3D graphics engines transmit data to and receives data from main memory 115 via memory controller 112.

**[0014]** In addition, MCH 110 includes a queue 114 to facilitate the interaction between memory 115 and memory controller 112. Queue 114 stores information (e.g., data, command information) from graphics accelerator 114 prior to the information being presented to memory 115. Although described herein with reference to a graphics accelerator/memory interface, one of ordinary skill in the art will appreciate that queue 114 may be implemented for other interfaces.

**[0015]** **Figure 2** illustrates one embodiment of queue 114. Queue 114 includes functional unit blocks (FUBs) 220. Queue 114 also includes control logic to facilitate the interface between graphics accelerator 113 and memory controller 112. In one embodiment, the die of MCH 110 is divided into two different partitions in order to compensate for the relatively large size of MCH 110. As a result, the functionality of queue 114 is divided between FUBs 1 and 2 to implement the partitioning of the MCH 114 die. This division makes it easy to remove restrictions on how the die should be partitioned and at the same time allows PV requirements to be met. In addition, dividing the queue the does not add extra gates, nor does it add any additional complexity.

**[0016]** According to one embodiment, FUB 1 is operated based upon a source clock domain, while FUB 2 is operated according to a destination clock domain. Consequently, data is loaded into queue 114 via a source clock, and is unloaded via a destination clock. Note that this scheme works for a queue 114 structure irrespective of the kind of clock crossing (synchronous or asynchronous). Further, there is uni-directional signaling between FUB 1 and FUB 2, such that there will be a strobe (put) and a packet associated with this put that flows from one FUB to the other. Consequently, there is no other combinatorial/boomerang that flows back in response to this.

**[0017]** **Figure 3** illustrates a detailed view of FUB 1 and FUB 2. FUB 1 includes logic associated with a load pointer and match logic. FUB 2 includes the storage elements to store data to be transmitted to memory controller 112 and an unload pointer. In addition, FUB 2 includes clock gating elements to gate the load pointer into the destination clock domain.

**[0018]** The load pointer indicates a location in the queue 114 storage elements in which to store information that is to written into the storage elements during a put command. Each time a put command is executed, the load pointer is incremented by control logic 250 to the next location in the queue to which information will next be written.

**[0019]** The unload pointer indicate a location in the storage elements in which information is to be read from during an unload command. Each time an unload command is executed, the unload pointer is incremented by control logic 250 to the next location in the queue from which information will next be read.

**[0020]** In one embodiment, the load pointer is clock crossed to the destination domain in FUB 1 to save a clock of latency. Similarly, the unload pointer may again be clock crossed to the source domain in the FUB 2. The data to be stored in queue 114 is directly flopped in the source clock domain in FUB 2. Since there is not much logic for this data (e.g., the data is from the output of a flop stage), there is no PV issue by flopping the data directly to FUB 2. Since the storage elements (e.g., latches or flops) are in FUB 2 the load pointer are decoded in the destination domain in order to determine the location in which to place the data.

**[0021]** In one embodiment, the clock crossed versions of the load pointer and the unload pointer are used to determine, at FUB 2, if there is a command present in queue 114. In a further embodiment, the availability of space in queue 114 is determined at the match logic within FUB 1 by using the load pointer and the clock crossed version of the unload pointer.

**[0022]** In one embodiment, the splitting of queue 114 into separate logic blocks enables the destination domain to be moved to a third FUB (FUB 3) at a later time. This move may be simply implemented by removing the destination logic block from FUB 2 and adding it to the hierarchy of FUB 3, and by making sure that the entities of the FUBs reflect these changes. If in this scenario it so happens that the logic block has been moved to another partition which is non-adjacent and if it leads to longer routing delays causing timing issues, the fix is as simple as adding an extra flop stage for the strobe and packet paths. .

**[0023]** The above-described mechanism enables the flexibility of moving the logic blocks involving queues around from one partition to another without having to spend an excessive amount of time having to redesign the queue logic. The mechanism also provides a potential of scalability, meaning fixing timing issues by simply adding flop stages on strobe and packet paths without any extra control logic. Further, the queue division works for high frequencies.

**[0024]** Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as the invention.